A Declarative Modeler for B-Spline Curves

Vincent Rossignol and Marc Daniel

Abstract. Declarative modeling aims at producing scenes or objects from the user's requirements, and be will briefly introduced. We will then present *MDC*, a Declarative Modeler for Curves, and the different ways for describing curves. We mainly focus on our internal model which allows us to simply manipulate B-splines curves preserving their properties.

§1. Introduction

The current geometric modelers make it possible to construct complex shapes. Nevertheless, the designer has to describe the studied objects by means of lists of coordinates, values or geometric primitives. This way of working, called imperative modeling, is often complex and tedious, even if the associated mathematical models are powerful.

Our goal through declarative modeling is to permit the creation of shapes by only providing a set of abstract specifications, generally based on geometric, topological or physical properties. The role of the computer is then to determine and/or explore the universe of shapes corresponding to the given description. This approach assumes that the description is not overconstrained. Moreover, the time used to describe the shape must be less than the time required to define it by manipulating control points. We are more interested with a "draft" than a very accurate result. A first attempt at declarative modeling of a B-spline curve, and preliminary concepts have been described in [1]. It has led to the new approach proposed in this paper. The method used for the initial description of the properties required by the designer is not very important in the current context, but must be as easy as possible. It can be found in [5]. Declarative modeling is made up of 3 stages:

(1) *the description stage*, where the user's description is transformed into an internal description,
(2) *the generation stage*, where the universe of solution(s) is constructed or sampled from the internal representation,
(3) *the presentation stage*, where solution(s) is (are) presented to the user.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 353–362.

ISBN 0-8265-1356-5.

The first stage is very important but will not be detailed here. It will be just introduced in the next section. References can be found in [4]. The generation stage transfers properties of a virtual curve into geometrical properties applied to the data of our mathematical B-spline model. We finally have to manage a set of control points and a set of geometrical constraints. The constraints link control points to the properties. We chose to focus the paper on the presentation of this model and explain how it is set up.

During the presentation stage, the user has the opportunity to browse through different sets of solutions and to select one of them. Moreover, he/her can interactively move the control points. But each control point can only be moved within a restricted region in which the geometrical constraints are checked.

§2. Curve Description

The description stage in our modeler can be done through 2 methods:

• **The natural description** consists of describing the properties of the curves via pseudo-natural language. For instance, a user can enter a description like: "My curve begins at the top bottom of my workspace", "it has a linear part in the middle", "it has two inflection points"... The description will be translated into a semantic graph that represents properties on the curves which is itself translated into our model presented in next section.

• **The visual description** is another way to enter properties on the curve. For handling properties on the curve, the user can visually insert properties with the mouse on the curve. Then, the computer will ask him for other information. For instance, suppose that we have a curve which corresponds to the natural description seen above. If the user wants to insert a cusp before the linear part, he just has to select this part and asks the computer to insert a cusp there. Then the modeler will ask for the right and the left tangencies. In this mode, work is directly achieved on the internal model.

§3. Constrained B-spline Curves

In this section, we will introduce the internal model for representing and manipulating the curves. It must have three properties:

(1) be as near as possible to the B-spline model,

(2) contain the constraints yielding the description,

(3) allow the user to manipulate the curve preserving the properties.

3.1 Preliminary definitions

Definition 1. *Let P be a point of $\mathbb{R}^2$. We can associate in a formal way a* function of constraint *$\overline{F}_P$ with P whose goal is to restrict any part Z of $\mathbb{R}^2$ according to a property $(\overline{F}_P(Z) \subset Z)$.*

Let C be a B-spline curve with $(n+1)$ control points. We actually are interested in functions of constraint for control points P_i. As these functions often have a generic formulation, it is sometimes convenient to replace the notation $\overline{F}_{P_i}$ with $\overline{F}(.,i)$. A function of constraint reduces the region associated to each constrained point:

Definition 2. *A* constrained point *$(\overline{P})$ is a triplet $(P, Z, \overline{\mathcal{F}})$ defined by:*
P, a point of $\mathbb{R}^2$,
Z, a convex subset of $\mathbb{R}^2$,
$\overline{\mathcal{F}}$, a set of functions of constraint applied to P.

We can now consider a B-spline applying this notion of constrained point. The control polygon is no longer a list of points, but a list of constrained points. It is named a constrained control polygon (next definition). So, with these assumptions, the functions of constraint are set up to ensure properties and to simply manipulate them. An example is proposed in the next section.

Definition 3. *Let C be a cubic B-spline. A* constrained control polygon *is a sequence of constrained points*

$$\overline{\Pi} = (\overline{P}_i)_{i\in\{0,1,\ldots,n\}},$$

with

$$\forall i \in \{0,1,\ldots,n\}, \overline{P}_i = (P_i, Z_i, \overline{\mathcal{F}}_i).$$

Definition 4. *Let $\overline{\Pi}$ be a constrained control polygon on a B-spline C. A constrained point $\overline{P}_i$ is called a* valid constrained point *iff*

$$\forall \overline{F} \in \overline{\mathcal{F}}_i, Z_i \subset \overline{F}(Z_i, i), \qquad (i.e. \quad Z_i = \overline{F}(Z_i, i)),$$

and $Z_i \neq \emptyset$ and $P_i \in Z_i$. If all the constrained points of $\overline{\Pi}$ are valid, $\overline{\Pi}$ is also said to be valid.

3.2 An example of functions of constraint

We choose a simple property "*a linear part on the curve*". We consider a cubic B-spline curve with a uniform knot vector. We define 4 functions called: $\overline{L}_{left}$, $\overline{L}_{midG}$, $\overline{L}_{midL}$ and $\overline{L}_{right}$. We apply these functions on 4 consecutive points of a constrained control polygon. $\overline{L}_{left}$ will be defined by

$$\overline{L}_{left}(Z_i, i) = Z_i \cap [P_{i+1}, \overrightarrow{P_{i+2}P_{i+1}}),$$

where $[P_{i+1}, \overrightarrow{P_{i+2}P_{i+1}})$ represents the ray defined by

$$\{P | \overrightarrow{P_{i+1}P} = k.\overrightarrow{P_{i+2}P_{i+1}}, k \in \mathbb{R}^+\}.$$

One can see in Figure 1 that $\overline{L}_{left}$ has been set up to reduce the region associated with a constrained point. In the same way, we can define $\overline{L}_{midG}$

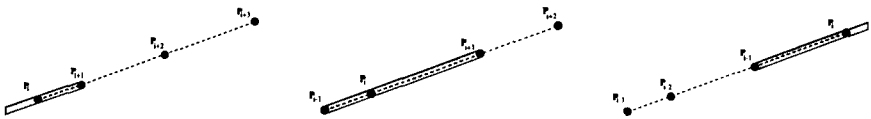

Fig. 1. $\overline{L}_{left}(\mathbb{R}^2, i)$, $\overline{L}_{midG}(\mathbb{R}^2, i)$ and $\overline{L}_{right}(\mathbb{R}^2, i)$.

and $\overline{L}_{midL}$ for the two middle points, and $\overline{L}_{right}$ symmetrically as $\overline{L}_{left}$, see Figure 2:

$$\overline{L}_{midG}(Z_i, i) = Z_i \cap [P_{i-1}P_{i+1}[, \quad \overline{L}_{midL}(Z_i, i) = Z_i \cap]P_{i-1}P_{i+1}],$$

$$\overline{L}_{right}(Z_i, i) = Z_i \cap [P_{i-1}, \overrightarrow{P_{i-2}P_{i-1}}).$$

We can now state the property for the "*linear part*"

Proposition 1 (Linear functions of constraint). *Let C be a cubic B-spline and $\overline{\Pi}$ a constrained control polygon on C. Assume that there exists $i \in \{0, 1, ..., n-3\}$ and*

$$\overline{L}_{left} \in \overline{\mathcal{F}}_i, \quad \overline{L}_{midG} \in \overline{\mathcal{F}}_{i+1}, \quad \overline{L}_{midL} \in \overline{\mathcal{F}}_{i+2}, \quad \overline{L}_{right} \in \overline{\mathcal{F}}_{i+3}.$$

We assume the knot vector to be uniform for the part of the curve associated with $\{P_i, P_{i+1}, P_{i+2}, P_{i+3}\}$. Then if $\overline{\Pi}$ is valid, the curve has a linear part defined by the line segment $[P_{i+1}, P_{i+2}]$.

Proof: The above assumptions imply that for a valid control polygon,

$$Z_i \neq \emptyset, P_i \in [P_{i+1}, \overrightarrow{P_{i+2}P_{i+1}}), \quad Z_{i+1} \neq \emptyset, \quad P_{i+1} \in [P_iP_{i+2}[,$$

$$Z_{i+2} \neq \emptyset, P_{i+2} \in]P_{i+1}P_{i+3}], \quad Z_{i+3} \neq \emptyset, \quad P_{i+3} \in [P_{i+2}, \overrightarrow{P_{i+1}P_{i+2}}),$$

so that points $P_i, P_{i+1}, P_{i+2}, P_{i+3}$ are aligned. We can also notice that this sequence satisfies

$$\overrightarrow{P_iP_{i+1}} = k.\overrightarrow{P_{i+1}P_{i+2}}, \quad \overrightarrow{P_{i+2}P_{i+3}} = k'.\overrightarrow{P_{i+1}P_{i+2}}, \quad (k, k') \in (\mathbb{R}^+)^2$$

Finally, the uniform knot vector yields that the line segment $[P_{i+1}P_{i+2}]$ is a part of C. □

This example is very convenient. But in the same way, we defined functions of constraint for properties like "*cusp*", "*tangencies*" etc.. An introduction to these constraints is available in [6]. This example emphasizes that an organisation for the functions of constraint exists as described in the next section.

3.3 Pieces of control polygon

For defining a linear part on a cubic B-spline, four points are required. Four functions are defined and inserted into the sets of the constrained control points. But these four points are not strongly linked. We introduce a structure linking points: this structure is called Piece of control Polygon (PcP).

Definition 5. *Let $\overline{\Pi}$ be a constrained control polygon on a B-spline curve C. A* Piece of control Polygon *is a triplet (I, m, BP) where*

- *I is the first subscript of the constrained control points associated with the PcP,*
- *m is the number of points associated with the PcP,*
- *BP is a bounding polygon for all the points of the PcP.*

We can now define "*typed PcP*". The type will depend on the property that the PcP handles. For example, a "*linear PcP*" will define a linear part on the curve. In such a case:

- I is the subscript of the first point associated to the linear PcP,
- $m = 4$,
- BP, a bounding polygon for the linear PcP.

Then, if we have

$$\overline{L}_{left} \in \overline{\mathcal{F}}_I, \quad \overline{L}_{midG} \in \overline{\mathcal{F}}_{I+1}, \quad \overline{L}_{midL} \in \overline{\mathcal{F}}_{I+2}, \quad \overline{L}_{right} \in \overline{\mathcal{F}}_{I+3},$$

the curve has a linear part located in bounding polygon BP. As discussed for the linear PcP, different types of PcP can be defined. We have currently implemented, among others, Inflection PcP (for inflection point), Convex PcP, Break point PcP. The curve can now be considered as a sequence of PcP.

Property 1 (Partition of the constrained control polygon). *Consider a constrained control polygon $\overline{\Pi}$ on a B-spline C and a PcP sequence on this polygon. The PcP sequence must partition $\overline{\Pi}$. In other words, each point is included in only one PcP. For a sequence $(PcP_i)_{i\in\{0,1,\ldots,n_{PcP}\}}$, we have $I_0 = 0$, $I_j + m_j = I_{j+1}$, for all $j \in \{0, 1, \ldots, n_{PcP} - 1\}$, and $I_{n_{PcP}} + m_{n_{PcP}} - 1 = n$.*

This property involves two statements. The first is that any point must be in a PcP. A point not pertaining to a PcP does not define any property, and is not required. The second is that a point belongs to only one PcP in order to ensure that no system of constraints on a point will be overconstrained. This approach does not lead to a theoretical minimum number of points defining a B-spline curve. Finally, we can say that the shape of the control polygon on the PcP is based on the shape preversing theorem stated in [2].

3.4 Convexity between two PcP

As the B-spline is defined on $\overline{\Pi}$ and not only on each PcP, B-spline curve segments exist, defined by points belonging to different PcP. These pieces, called "neutral parts", must not introduce unexpected properties. Each neutral part is defined with a control polygon Q_i of four points. In order to avoid unexpected properties in a neutral part, 4 functions of constraints, called joining

functions, are added to preserve the convexity of polygons Q_i. They work similarly to the linear functions seen in Section 3.2. For any PcP, an orientation of the curvature can be associated with each tip of the corresponding curve segment (clockwise or underclockwise). For two consecutive PcP, the orientation of the curvature, called signature (signature *sig* is 1 or -1), must be the same for both neighbouring tips, so that there is no possible inflection on a neutral part.

3.5 Constrained B-spline

Definition 6. *A* constrained B-spline $\overline{C}$ *is defined by*

- k, *its order (currently equal to 4 - i.e.,cubic B-splines)*,
- $n+1$, *the number of its constrained control points*,
- T, *its knot vector*,
- $\overline{\Pi}$, *its constrained control polygon*,
- n_{PcP}, *the number of PcP in* $\overline{\Pi}$,
- PcP, *the sequence of the Pieces of control Polygon*,
- $(sig_i)_{i\in\{2,\ldots,n_{PcP}\}}$, *the signs for the curvature of the neutral parts.*

The first three components correspond to the classical components of a B-spline curve. The knot vector is considered uniform ($t_{n+1} = t_n + 1$) except when a cusp is required on C. In this case, we need to increase the multiplicity of one knot up to 3. For example, if we need to increase the multiplicity at t_p:

$$t_{p-1} = t_p = t_{p+1}, \quad t_{p-2} = t_p - 2, \quad t_{p+2} = t_p + 2.$$

Otherwise, relation $t_{i+1} = t_i + 1$ is preserved.

Definition 7. *Given a constrained B-spline* $\overline{C}$, $\overline{C}$ *is said to be* valid *if its constrained control polygon is valid. This definition is very important. If a constrained B-spline is valid, all the properties imposed through the functions of constraints are checked. We will also see in the next section that a point is allowed to move within its associated region, preserving these properties.*

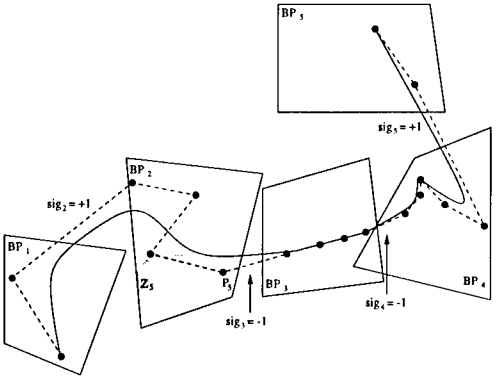

Fig. 2. A complete example of a constrained B-spline.

A complete example of a constrained B-spline curve is illustrated in Figure 2. Three properties are required (an "*inflection point*", a "*linear part*" and a "*break point*"). Region Z_5 is the region where point P_5 can move. All locations in this region preserve the inflection point and the linear part. These locations also ensure that no inflection will be created between the second and the third PcP. Notice that the first and the last PcP are just here to begin and to end curve, and do not have exactly the same behavior as the others.

We defined a function δ which allows us to compute all the regions of the constrained control polygon. For all constrained points, this function initializes the associated region to $\mathbb{R}^2$ and applies all the constrained functions to the region. We can finally state:

Property 2 (Move a point within its region). *Let $\overline{C}$ be a valid B-spline, and i a subscript of a constrained control point ($0 \leq i \leq n$). For all the positions of P_i in Z_i, the properties associated with the functions of constraints are checked.*

This property is one of the most important in the model. When a point is moved within its associated region, a new valid constrained B-spline can be obtained by applying function δ to this new B-spline.

§4. Choosing an Initial Curve

The model we introduced can obtain the different solutions to the designer's problem. Nevertheless, a first curve has to be computed. This section introduces the main steps of this construction algorithm. To solve the problem, we assume that a sequence of PcP is given (this sequence has been constructed during the description stage which is not presented here). Defining the first curve consists in finding a position for all control points so that the application of function δ leads to a valid constrained B-spline. The construction algorithm is divided into 3 stages:

1) Choice of the signature vector,
2) Initialization of the regions,
3) Pick of the control point locations.

4.1 Choice of the signature vector

The signature vector describes the curvature between each PcP. As described in Section 3.4, it is composed of $n_{PcP} - 1$ values in the set $\{+1, -1\}$. All the configurations are not correct. For choosing the values in the vector, we use relations depending on the types of the PcP (see Figure 3). When all the relations between the entries of the vector are defined, an instance of these entries are looked for. The solution is generally not unique. In such a case, different families of solutions have to be investigated (see Figure 4). It may happen that the whole set of relations is inconsistent. This corresponds to an inconsistent description of the curve (for example, "*a closed curve with only one inflection*") and no curve can be computed.

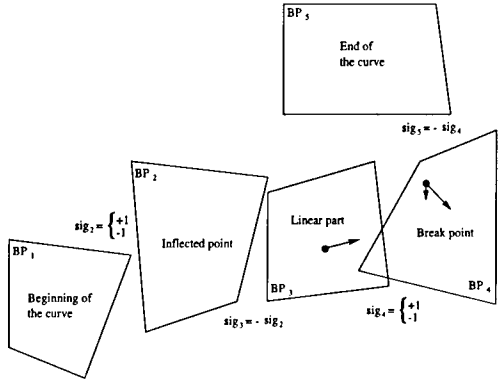

Fig. 3. Relations between signatures.

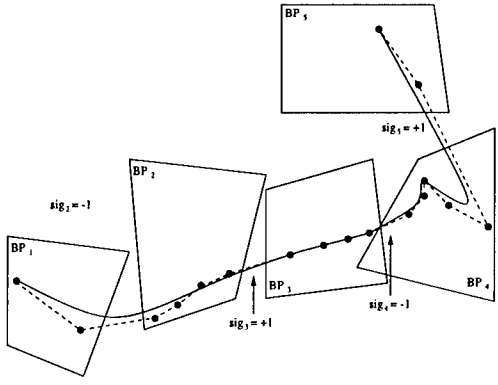

Fig. 4. Another solution than those proposed Figure 2.

4.2 Initialization of the regions

The initialization stage formally sets all the regions to $\mathbb{R}^2$, and reduces them to the bounding polygon of the corresponding PcP. The reduction is then obtained by a geometric construction for each function of constraint, one function at a time. It may happen that an empty region is produced: the description of the curve is inconsistent.

4.3 Choice of the control point location

The method cannot be detailed here. It is divided in two stages:

1) The location of the external (*i.e.*, the first and the last) points of each PcP is determined,
2) The location of the internal points (*i.e.*, all others) are computed.

The first stage is achieved with a specific algorithm. The locations of the first two external points are choosen. An attempt to find the location of

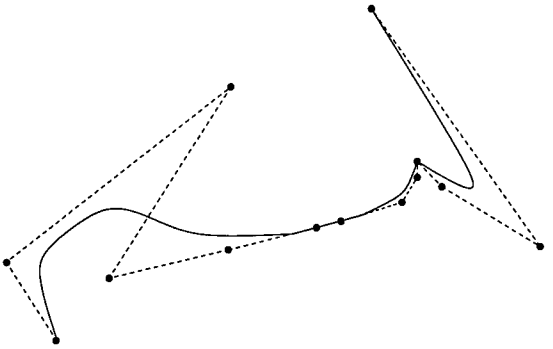

Fig. 5. The curve of Figure 2 defined with less control points.

the next point is made. If no valid location is found, backtracking is started. After a given number of failures, we claim that no solution can be computed, without determining if there is no solution or if we are unable to find it. But generally, a result is produced when it exists. The locations of the internal points are deduced from specific heuristics which never fail.

Once the location of all the control points are computed, applying function δ provides the valid constrained B-spline which can be now manipulated.

§5. Curve Improvement

In order to produce more interesting curves, final improvements have to be applied on the constrained curves. They mainly concern the quality of the control polygon which takes into account the spatial distribution of points, and the reduction of the number of control points.

The quality of the control polygon is defined through a measure of quality (result in interval $[0, 1]$). An increase of the quality is obtained by moving the contrained control points one by one.

As we already mentioned, the number of control points can be too large. Decreasing this number is important while preversing the shape of the curve. General results have been proposed in [3]. As the important properties on the curve and the control points handling these properties are known, our algorithm is easier: first remove non-critical points for the properties, then optimize the distance between the first curve and the reduced one. An example is shown in Figure 5.

§6. Conclusion

MDC validates the approach described in this paper. Improvements of the program are still necessary, but it already provides interesting results. A declarative modeler does not exclude a classical modeler but can provide a way for the user to eliminate the most tedious part of the design process.

The declarative approach has another application: the produced curve can be considered as a classical B-spline. Its properties can be kept so that semantic information is available (which is not so far from form features in CAD). This information would be useful in applying other algorithms afterwards to the curve.

References

1. Daniel, M., Declarative Modeling of fair shapes: An additional approach to curve and surface computations, in *Advanced Course on FAIRSHAPE* (J. Hoschek and P. Kaklis - eds.), B. G. Teubner Stuttgart, 1996, 77–85.
2. Kantorowitz, J. E. and Y. Schechner, Managing the shape of planar splines by their control polygons, Computer-Aided Design **25-6** (1993), 355–364.
3. Schumaker, L. L. and S. S Stanley, Shape-preserving knot removal, Computer Aided Geometric Design **13** (1996), 851–872.
4. Desmontils, E., and J. Y. Martin, Properties Taxonomy in Declarative Modeling, CISST'97, Las Vegas, 130–138.
5. Rossignol, V., and M. Daniel, Le module de description d'un modeleur déclaratif, Research Report IRIN number 177, 1998.
6. Rossignol, V., and M. Daniel, Étude des contraintes dans la phase de génération d'un modeleur déclaratif de courbes, Research Report IRIN number 178, 1998.

Vincent Rossignol
IRIN (Institut de Recherche en Informatique de Nantes)
2, Rue de la Houssinière, BP 92208
44322 Nantes Cedex 3, France
Vincent.Rossignol@irin.univ-nantes.fr

Marc Daniel
XAOlab-ESIL/LIM
Campus de Luminy, case postale 925
13288 Marseille cedex 9, France
Marc.Daniel@esil.univ-mrs.fr